Eigenfeatures for Planar Pose Measurement of Partially Occluded Objects

John Krumm
Intelligent Systems and Robotics Center
Sandia National Laboratories
MS 0949, P.O. Box 5800
Albuquerque, NM 87185
jckrumm@sandia.gov

Abstract

Planar pose measurement from images is an important problem for automated assembly and inspection. In addition to accuracy and robustness, ease of use is very important for real world applications. Recently, Murase and Nayar have presented the "parametric eigenspace" for object recognition and pose measurement based on training images. Although their system is easy to use, it has potential problems with background clutter and partial occlusions. We present an algorithm that is robust in these terms. It uses several small features on the object rather than a monolithic template. These "eigenfeatures" are matched using a median statistic, giving the system robustness in the face of background clutter and partial occlusions. We demonstrate our algorithm's pose measurement accuracy with a controlled test, and we demonstrate its detection robustness on cluttered images with the objects of interest partially occluded.

1. Why Work on Planar Pose Measurement?

Planar pose measurement is an important part of automated assembly and inspection. In this paper, we are envisioning a workcell of the type illustrated in Figure 1. An overhead camera is pointed down at objects resting on a flat surface. The task is to measure the pose, (x, y, θ), of a given object in the image.

Although there are several solutions available to the pose measurement problem, both commercially and academically, none of the solutions have yet to win widespread appeal. One of the main barriers to increased use of computer vision in automated manufacturing is that the vision systems are difficult to tune. Pose measurement is intended to function as part of an automated manufacturing line. But this advantage is lost when the vision system requires reprogramming from a skilled operator to account for changes in illumination, optics, and objects. Current commercially available solutions typically require a training phase in which an operator manually helps the vision system identify important features of the objects of interest. These features must be carefully chosen based on their consistency and their ability to indicate the pose of the object. This requirement leads to heuristic rules for the operator to follow such as "specular highlights are bad features because they shift based on the orientation of the object, lights, and camera," and "line segment features are not good for localizing the object along the line." Given the cost of computer vision experts, it often appears less expensive to find less flexible or more labor intensive solutions to the pose measurement problem.

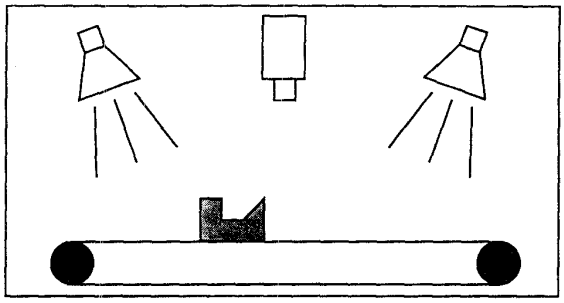


Figure 1: A typical pose measurement station with an overhead camera and lights pointed down at a conveyor belt carrying objects to be assembled or inspected.

In this paper, we present a new solution to the pose measurement problem. It is based on Murase and Nayar's "parametric eigenspace" idea[2], which uses principle component templates based on training images. We show how to apply this idea to multiple, automatically detected features on the object. We match features using a median distance measure, which gives the algorithm robustness. Using features instead of monolithic templates, our algorithm overcomes problems of segmentation, background clutter, and partial occlusions, while retaining the automatic programming advantage of the original system.

2. Parametric Eigenspace

A new technique for pose measurement, called "parametric eigenspace", has been developed by Murase and Nayar [2]. Their work is related to earlier eigenface research by Turk and Pentland [7]. The method is used to recognize objects and measure their orientation based on training images of the objects in different orientations. This solution is attractive because it requires no expert human assistance for picking features. A brief explanation of parametric eigenspace follows.

In its full form, as explained in [2], the parametric eigenspace method works on a presegmented image to identify an object and give its orientation around one axis under a few

This work was performed at Sandia National Laboratories and supported by the U.S. Department of Energy under Contract DE-AC04-94AL85000.

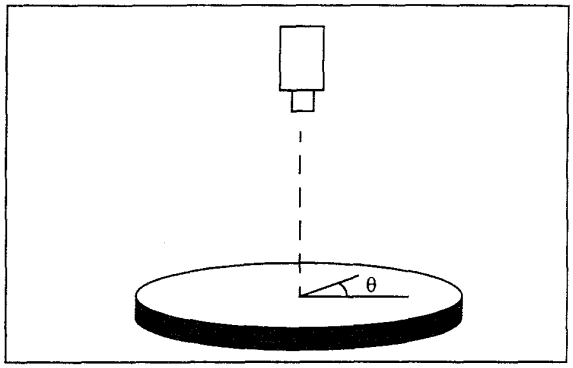


Figure 2: Training images are taken with an overhead camera looking down at the object which rests on a motorized rotation table.

different lighting conditions. The method is based on a series of training images of the object taken at different orientations. In our implementation, we use 90 or 180 training images of the object (four degrees or two degrees apart) taken from a camera directly overhead while the object is rotated on a motorized table below. This setup is illustrated in Figure 2.

For training, the object is first segmented from each training image into a rectangular image patch. Segmentation in the training phase is normally easy because the scene can be carefully controlled. We use a backlit table to make a binary mask of the object. The segmented patches must be the same size for each training image. This equal size requirement means the patches must include significant parts of the background if the object is elongated or has significant concavities, as illustrated in Figure 3.

Each segmented training patch is normalized in intensity by dividing each pixel by the sum of the squares of the pixels in the unnormalized patch. While Murase and Nayar also normalize for size, we do not since we assume the camera will always be the same distance from the object. Finally, the mean of the entire normalized set of training patches is computed and subtracted from each normalized patch. Each of these processed patches is scanned in raster order to form a column vector containing all the pixels in the patch. All the column vectors are placed side by side into a matrix X. The sample covariance matrix is formed as

$$Q = XX^T.$$

The eigenvectors of Q form an orthogonal basis set for the normalized, zero mean training patches. These eigenvectors are called eigenimages when they are scanned from column vectors back into images. The normalized, zero mean training patches can be expressed as a weighted linear sum of the eigenimages. Moreover, the patches can be accurately approximated by only the first few eigenimages. In equation form, this is

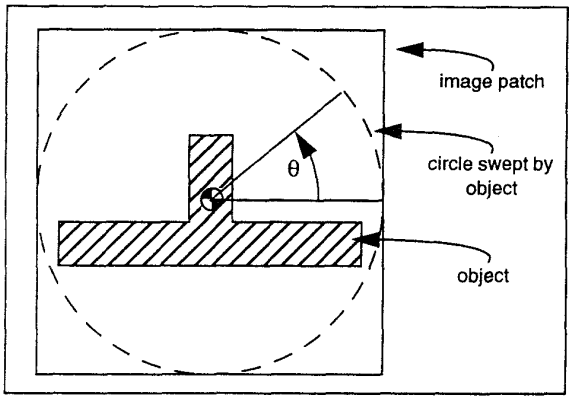


Figure 3: In previous approach, segmented image patch must be large enough to contain all trained orientations of object, which could include large parts of background.

$$\bar{p}_i \approx \sum_{j=0}^{l-1} c_{ij}\bar{e}_j \qquad (1)$$

where

$\bar{p}_i$ = i^{th} normalized, zero mean training patch

$\bar{e}_j$ = j^{th} eigenvector of Q

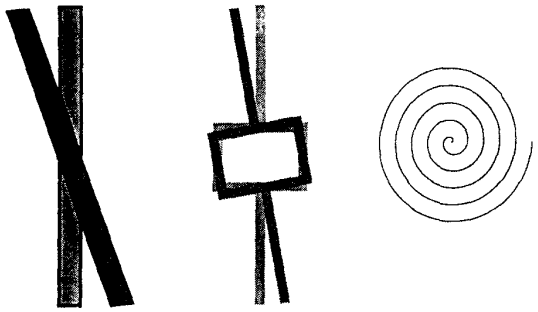
c_{ij} = weighting coefficient = $\bar{p}_i \cdot \bar{e}_j$

l = number of eigenimages used

The simple computation of the weighting coefficients comes from the fact that the eigenvectors are orthogonal. For our experiments, we used $l = 10$.

Equation (1) implies that each training patch $\bar{p}_i$ can be represented by l coefficients $(c_{i,0}, c_{i,1}, ..., c_{i,l-1})$. These coefficients are a point in an l-dimensional space, and each training patch projects to such a point. Since the training images represent an ordered progression of angles (θ in Figure 2), the coefficients plotted in l-dimensional space normally fall on a smooth curve. Each point represents a different training image and thus represents a different orientation of the object. This curves resides in "parametric eigenspace", because it can be parameterized by the angle θ.

Pose measurement in Murase and Nayar's formulation consists of first segmenting an input image by some means to find the object. This rectangular image patch is then normalized and the mean of the training patches is subtracted (same processing as training images). This processed patch is projected into the parametric eigenspace by taking dot products with the l previously computed eigenimages. To find the angle θ, the parametric curve is interpolated to find the closest point. More details can be found in [2], which also discusses a similar approach to object recognition and



small overlapping region | bounding box of overlap includes background | possibly no overlap

Figure 4: It is difficult to make good templates from the overlapping regions of these objects' silhouettes.

training under different illumination.

The advantage to the eigenspace approach is that the training is automatic. One disadvantage is that the object must first be segmented in the run-time image, which can be difficult in any image without a uniform background. Another disadvantage is that the basic parametric eigenspace method works on rectangular image patches that contain the whole object. Thus, for optimum matching, none of the object can be occluded, and the background of the object to be detected in an image cannot be different from the background in the training images. Rectangular image patches become a problem when the object does not have a generally rectangular shape, because other objects can intrude in the background. Our algorithm addresses all these problems.

Recently, Murase and Nayar have modified their approach to address the problems of segmentation and background clutter[3]. Their "image spotting" algorithm scans the image for the object, which solves the segmentation problem. (This assumes that the object's appearance is invariant with respect to translation in the image, which is approximately true when using a long focal length lens.) The algorithm uses training images that are reduced in size to the area of common overlap between the object's silhouettes in the training images at different angles, thus eliminating the background from the training images. Some objects do not have much overlap in their silhouettes, so the image spotting algorithm can sometimes split the training images into subsets that have sufficient overlap. Thus, the reduced size training templates solve the problems of background clutter for some objects, but not all. Some objects defy this splitting, as their silhouettes have virtually no overlap that could be contained in a bounding box without background as the object rotates in discrete increments. Three illustrations of such shapes are given in Figure 4. The problem of partial occlusions still remains, too.

3. Eigenfeatures for Pose Measurement

We have developed a new way of using parametric eigenspace that avoids the problem of segmentation, background clutter, and partial occlusions. We avoid the segmentation requirement by applying our method at every offset in the image, using the fast Fourier transform to speed up the projections into eigenspace. We solve the problem of background clutter and partial occlusions by using several small, rectangular image patches on each object rather than one large patch. In addition, using features means our algorithm does not use large, uniform regions of the object for matching, which often leads to false matches. Our method retains the advantages of the original algorithm in that it works entirely based on training images and requires no programming from a skilled operator. The remainder of this section explains our method in detail.

3.1 Gathering Training Features

We gather training images using an overhead camera pointed down at a backlit table mounted on a motorized rotator. We take images every two or four degrees using overhead lighting, giving a total of 180 or 90 training images per object. We take another set of images at the same set of angles with the overhead lights off and the backlit table on. The backlit images are thresholded to form masks for segmenting the training images. This backlit segmentation is used only for training and is not part of the on-line system.

Features consist of small, rectangular image patches. We use a feature size of 15 x 15 pixels along with a feature-finder developed by Shi and Tomasi[6]. Given an image, their algorithm produces a list of rectangular image patches of a prespecified size that is ranked based on the features' ability to be tracked through image sequences. We find their features to be good for pose measurement, too, because they are good at localizing position, *e.g.* points and corners. Shi and Tomasi's recipe for finding good features is to first compute partial derivatives at every pixel in the image. If the image is $I(x, y)$, then the partial derivatives are $I_x(x, y)$ and $I_y(x, y)$. For the numerical derivatives, we use Savitzky-Golay filters as described in *Numerical Recipes* [4] with the derivative filter size equal to the window width (15 x 15). At each pixel, a matrix is formed whose elements are sums of the products of the partial derivatives taken in the feature windows. The matrix is

$$Z = \begin{bmatrix} \sum I_x^2(x, y) & \sum I_x(x, y) I_y(x, y) \\ \sum I_x(x, y) I_y(x, y) & \sum I_y^2(x, y) \end{bmatrix},$$

where the sums are taken over the pixels in the feature window. The second eigenvalue of this matrix is used to rank the feature - the higher the better. The top 30 features of a wire connector at different angles are shown in Figure 5.

Shi and Tomasi's algorithm is especially convenient, since the only parameters it takes are the size and number of fea-

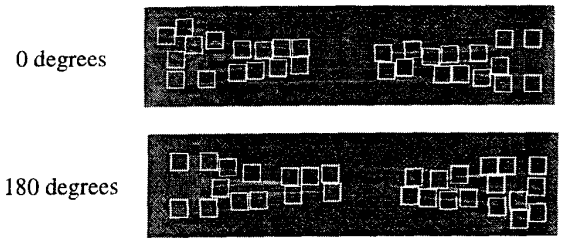


Figure 5: Features automatically selected on wire connector training images.

tures. We constrain the features to have no overlap, and to be centered somewhere within the mask computed from the backlit images.

Every training image has its own set of features - there is no tracking of features from image to image. Thus, features on certain parts of the object, such as specular highlights, could come and go as the object rotated. In a more general 3D orientation problem, rotating the object could cause parts of the object to come into and out of view, making feature correspondence impossible. Since we do not require feature continuity from angle to angle, this would not be a problem for our algorithm.

3.2 Training

The training for our method begins in the same way as described for the image patches in Section 2, except it uses small features instead of image patches. For n training images and m feature per image, we have mn total features. Since we do not track features through the training images, we must have some other way to account for the changing appearance of a given feature due to rotation. We do this by noting how the pixels in each feature window change as the object rotates slightly beneath them. For a given feature in training image i, we take out a feature in the same (x, y) location in training images $i-1$ and $i+1$ (with circular wrap-around of the indices on the first and last training images). This gives a total of $3mn$ features for each object. Each of these features is processed as described for the image patches in Section 2 (normalizing and subtracting aggregate mean).

The $3mn$ processed features are scanned in raster order into column vectors. We designate the pixels from the k th feature from image i to be $\bar{v}_{ik}$, where $i \in [0, 1, ..., n-1]$ indexes the n training images (and therefore the orientations θ_i), and $k \in [0, 1, ..., m-1]$ indexes the m features. The two features in the same location from images $i-1$ and $i+1$ are designated $\bar{v}^-_{ik}$ and $\bar{v}^+_{ik}$ respectively. If the dimensions of the features is w x w pixels, then each feature vector will be w^2 x 1.

The processed feature vectors are assembled into the columns of matrices:

$$X = [\bar{v}_{00} \mid \bar{v}_{01} \mid ... \mid \bar{v}_{ij} \mid ... \mid \bar{v}_{(n-1)(m-1)}]$$

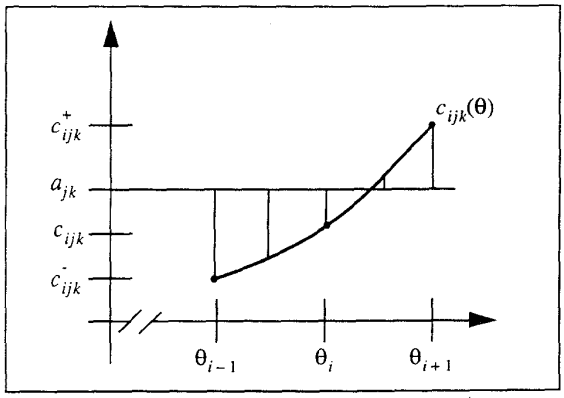


Figure 6: Parametric eigenspace for one eigenimage of one feature at one angle. We compute the distances between the parametric curves and actual eigenvector weights a_{jk} to find the best angle.

$$X^- = [\bar{v}^-_{00} \mid \bar{v}^-_{01} \mid ... \mid \bar{v}^-_{ij} \mid ... \mid \bar{v}^-_{(n-1)(m-1)}]$$

$$X^+ = [\bar{v}^+_{00} \mid \bar{v}^+_{01} \mid ... \mid \bar{v}^+_{ij} \mid ... \mid \bar{v}^+_{(n-1)(m-1)}] .$$

The order of the vectors in these matrices is irrelevant. (The bars in the matrices above represent matrix partitioning and not absolute value.) The dimensions of X, X^-, and X^+ are w^2 x mn. The sample covariance matrix of all $3mn$ features is formed as

$$Q = [X \mid X^- \mid X^+] [X \mid X^- \mid X^+]^T ,$$

whose dimensions are w^2 x w^2. In our implementation, $w = 15$, so the size of Q is a relatively modest 225 x 225.[1] We compute the eigenvectors of Q and designate them as $\bar{e}_j$, where $j \in [0, 1, ..., l-1]$ indexes the l eigenvectors that we use. In our implementation we use 10 eigenvectors. We call the $\bar{e}_j$ "eigenfeatures".

Each feature can be approximated as a weighted linear sum of the first few eigenvectors. These weights are $c_{ijk} = \bar{v}_{ik} \cdot \bar{e}_j$, $c^-_{ijk} = \bar{v}^-_{ik} \cdot \bar{e}_j$, and $c^+_{ijk} = \bar{v}^+_{ik} \cdot \bar{e}_j$. With these coefficients, we form small parametric eigenspaces as quadratics fit through sets of three parameterized points $(\theta_{i-1}, c^-_{ijk})$, (θ_i, c_{ijk}), and $(\theta_{i+1}, c^+_{ijk})$ for all images i, eigenvectors j, and features k. Figure 6 shows a sample quadratic for one eigenvector of one feature in one image. We call this function $c_{ijk}(\theta)$.

3.3 Detection and Pose Estimation

Our algorithm scans the input image in order to find the features on the object. An image to be analyzed must be processed in the same way as the features in the training image.

1. There is a technique outlined in [1] that shows how to compute the eigenvectors of XX^T by instead computing the eigenvectors of X^TX. If X is tall and narrow, this leads to a smaller eigenvector problem. In our case, however, X tends to be wider than it is tall, so we compute the eigenvalues of XX^T directly.

In order to normalize each feature-sized patch, we compute the local power at every point in the image in w x w windows by convolving a w x w, unit-height rectangle function with an image where each pixel has been squared. Each pixel in the image is considered as the center of a feature. Overlapping w x w, feature-sized windows in the image are projected onto the eigenimages. The projections are computed as correlations between the normalized input image and the eigenvectors. Both the convolution and correlations are done in the Fourier domain for speed. We also subtract the mean of all the training features as appropriate. This processing leaves us with an image where each pixel contains l eigenvector coefficients a_j.

To find the object and estimate its pose, we scan through the image in small increments (increments of one to five pixels) looking for the appropriate features in the appropriate spatial configuration. At each point we check all n training angles. For a given image point and a given training angle θ_i, we consider the point to be centered on the first feature ($k = 0$) of the m features for that angle. The centers of the other features are picked up in the image at the appropriate offsets with respect to the first feature. The eigenvector coefficients at these feature points are called a_{jk} where $j \in [0, 1, ..., l-1]$ indexes the l eigenvectors and $k \in [0, 1, ..., m-1]$ indexes the m features. To get a rough estimate of quality of the match at θ_i, we compare the image features to the trained features with no interpolation between angles. We compute a squared distance for each feature as

$$d_{ik} = \sum_{j=0}^{l-1} [c_{ijk} - a_{jk}]^2,$$

where the sum is taken over the l eigenvectors. If we have the approximate correct location in the image and the approximate correct angle index i, then all the d_{ik} will be small. If we have the approximate pose but the object is partially occluded, then only some of the d_{ik} will be small, because only some of the features will be visible. Therefore, we use the median to combine the d_{ik} into a single distance measure:

$$d_i = \text{median}_k (d_{ik}) .$$

The rough pose estimate is the position and angle that give the minimum d_i. The resolution of this estimate is limited to the pixel resolution of the image scan in location and the angle increment of the training images in orientation. We designate the best angle index as i^*.

Given the rough pose estimate, we refine the position and orientation with a gradient descent search. We form a distance function $d_{i^*}(\theta)$ that combines the distances between the lm eigenvector coefficients and the corresponding parametric eigenspace quadratics for the rough training angle estimate θ_{i^*}:

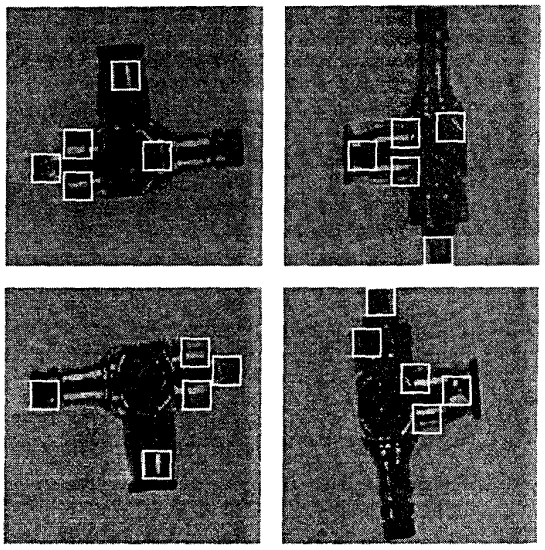


Figure 7: Shiny metal valve used for testing. Automatically found features are shown in rectangles. Specular highlights change depending on the valve's angle, meaning that consistent features are difficult to find.

$$d_{i^*}(\theta) = \sum_{k=0}^{m-1} \sum_{j=0}^{l-1} [c_{i^* jk}(\theta) - a_{jk}]^2,$$

where the sums are taken over the m features and l eigenvectors. One of the addends in the sum is illustrated in Figure 6. Since $c_{i^* jk}(\theta)$ is a quadratic in θ, $d_{i^*}(\theta)$ is a quartic, and

$$\frac{dd_{i^*}(\theta)}{d\theta} = 0$$

is a cubic. One of the solutions to the cubic minimizes the sum of squared distances to give the best angle θ. We wrap this closed form minimization in a gradient descent over pixel location to give the final subpixel pose estimate. We do not use a detection threshold since we assume the object is present somewhere in the image.

We were recently made aware of a similar approach to this problem developed by Ohba and Ikeuchi. Like us, they use a principle component analysis of features taken from the object. The major differences are that they employ a step to eliminate similar-looking features, and they use a voting scheme to find the object rather than the image scanning that we use.

4. Results

We tested our algorithm for both accuracy and robustness. For accuracy, we used a shiny, metal valve as the object, shown in Figure 7. The mirror finish on this object meant that the features consisted mostly of specular highlights,

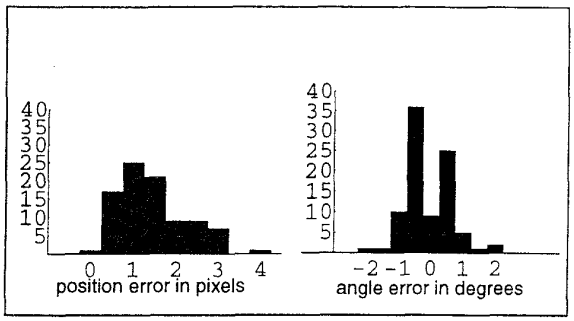


Figure 8: Histograms of position and angle errors for 90 test images of valve.

which changed dramatically as the object rotated. This would confuse any algorithm that depended on tracking features. We took 90 training images four degrees apart and used five features. We tested the algorithm on 90 test images taken at orientations halfway between each training image. We scanned the image in one-pixel increments. Our algorithm found the object in every image, with an average error in position of 1.4 pixels with a standard deviation of 0.8 pixels, and an average absolute value error in angle of 0.6 degrees with a standard deviation of 0.3 degrees. Histograms of the errors are in Figure 8.

Our test for robustness used two different objects - a long, thin wire connector and a shiny, metal pipe connector shaped like a "T". The wire connector has very little overlap between its silhouettes as it rotates, making a single template nearly impossible to use for this object. The "T" connector's shininess makes it difficult to track features on it. For both these objects, we used 180 training images taken two degrees apart and 30 features. We scanned the image first in increments of three pixels and then increased the search resolution to one pixel centered around the best result from the first pass. We tested the algorithm on images with background clutter and partial occlusions. The algorithm correctly found the object in about 80% of the test images. Successful results are shown in Figure 9.

5. Conclusion

We have shown how to use eigenfeatures for pose measurement in the plane. The use of training images to find good features makes the algorithm work without a skilled operator. The eigenvalue decomposition makes the algorithm more efficient than raw pixel matching. By scanning the image for the object, we avoid the problem of segmentation. The innovative use of features rather than monolithic templates allows the algorithm to work in spite of background clutter and partial occlusions. The combination of eigenspace analysis and features provides for a simple, accurate, and robust solution to the planar pose measurement problem.

References

[1] Murakami, Hiroyasa and B.V.K. Vijaya Kumar. "Efficient Calculation of Primary Images from a Set of Images", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(5), September 1982, 511-515.

[2] Murase, Hiroshi and Shree K. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance", *International Journal of Computer Vision*, 14(1), 1995, 5-24.

[3] Murase, Hiroshi and Shree K. Nayar, "Image Spotting of 3D Objects using Parametric Eigenspace Representation", *9th Scandinavian Conference on Image Analysis*, June 1995, 325-332.

[4] Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C (Second Edition)*, Cambridge University Press, 1992.

[5] Ohba, Kohtaro and Katsushi Ikeuchi. "Recognition of the Multi Specularity Objects using the Eigen-Window". Carnegie Mellon University School of Computer Science Technical Report CMU-CS-96-105, February 29, 1996.

[6] Shi, Jianbo and Carlo Tomasi. "Good Features to Track", *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994, 593-600.

[7] Turk, Matthew and Alex Pentland. "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, 3(1), 1991, 71-86.

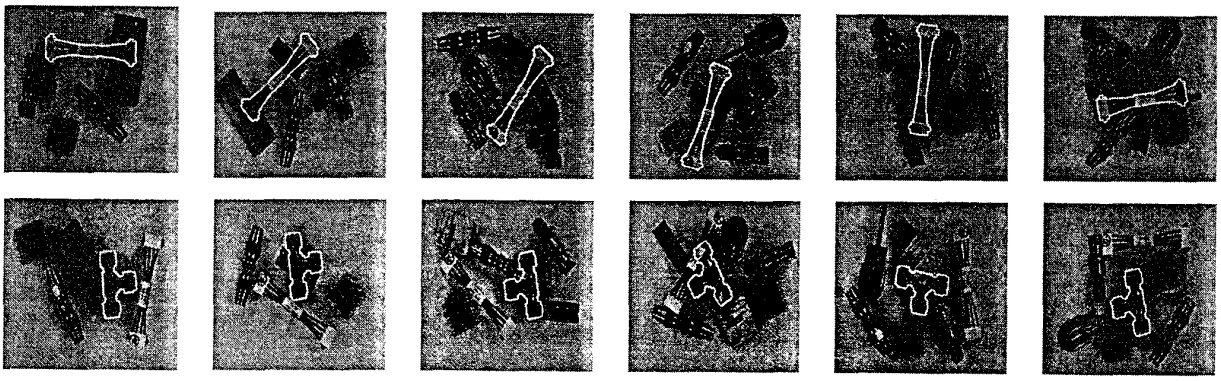


Figure 9: Results of object detection on images of partially occluded wire connector (top row) and "T" connector.